


CS II topics

I. Expanded review of CS I

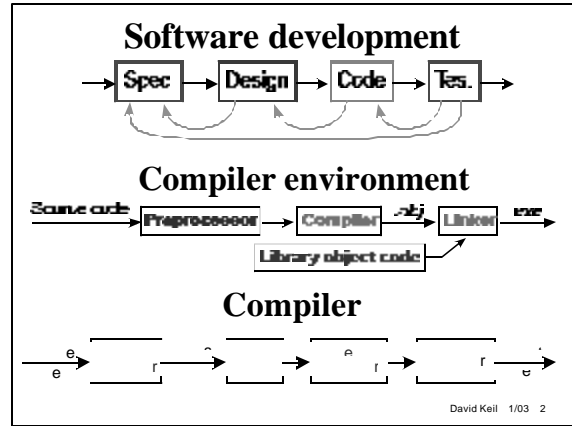
II. Compound types and collections

III. Applied knowledge




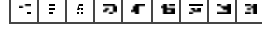

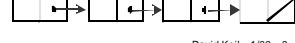
1. Standard types, control structures, subprograms
2. Structure types, classes
3. File input/output
4. Arrays
5. Sorting and searching
6. Pointers, dynamic allocation, lists
7. Software development
8. Languages, language processing
9. CS and discrete probability



David Keil 1/03 1



Some CS II topics in diagrams

- Function 
- Module hierarchy 
- Structure 
- Array or random-access file 
- Collection 
- Linked list 

David Keil 1/03 3

- ### Some themes of CS II
- Defining a *class* lets us model the objects found in a problem domain
 - A *collection* class lets us model a group of similar objects
 - We implement a collection using an *array* of objects, or a *linked* structure
 - *Algorithm analysis* helps manage time complexity of processes
 - *Software engineering* helps manage complexity of developing a solution
- David Keil 1/03 4

- ### Features of programs we will work with
- Longer than in CS I
 - We may need to focus on one part of a program at a time
 - Modularity is unavoidable
 - Algorithm choice, procedural abstraction, and data abstraction are evident in the design
- David Keil 1/03 5

- ### Sequential and random access
- Stream and list data is *sequential***
- Access is from start to end
 - *Examples:* keyboard (*cin*), screen (*cout*), text file
- An alternative is *random access***
- Program may store or retrieve an arbitrary item of a collection
 - *Examples:* random-access file; array; vector
- David Keil 1/03 6

Data structures

Interfaces

- General collection • Dictionary
- Matrix • Stack
- Queue

Implementations

- Array (linear or two-dimensional)
- Dynamically-allocated linked list

David Keil 1/03 7

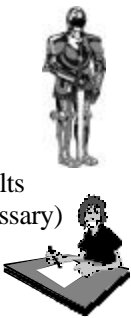
Basic skills expected

1. Write code to implement a flowchart
2. Solve a problem that requires a nested loop
3. Define subprograms with parameters
4. Define a structure type
5. Perform simple operations on strings
6. Perform file I/O
7. Trace a simple program to find and fix bugs

David Keil 1/03 8

Software quality issues

- User interface must be clear and straightforward
- Robust (“bullet-proof”) software is ready to handle any error condition
- Program should give correct results (including error messages if necessary) on *any* input
- Ensuring quality begins at the specification stage



David Keil 1/03 9

Debugging

- Syntax errors – Know the language!
- Logic errors
 - Output is wrong
 - What *intermediate value* is wrong?
 - *Tracing* is the basic tool
 - Know the language!
 - Use deduction like a detective
 - Expect surprises

David Keil 1/03 10

Applied-knowledge topics

6. *Searching and sorting* algorithms to manipulate arrays
7. *Software development issues*: Professional ethics, large projects, choices of tools
8. *Languages and language processing*: Procedural and other languages, interpreting vs. compiling; lexical analysis; parsing
9. *CS and discrete probability*: Probability theory relates to applications such as games, average-case analysis

David Keil 1/03 11

Semester project

- You will build a *collection* (database), using an *array of structures*
- The main features and techniques presented in the course will be used in the project
- Project will be completed *by stages*
- Other aspects: menu-driven application, file maintenance, searching and sorting of records
- *Documentation, testing, and debugging* are key aspects of the project

David Keil 1/03 12